projetos típicos de programação de grande porte desenvolvidos nos Estados Unidos, segundo Boehm. Observa-se ali que atualmente uns 80% dos custos de desenvolvimento de um sistema de programação vão por conta do *software*, ou seja, essencialmente dos analistas e programadores. Até 1985 prevê-se que essa relação atingirá quase 90%.

Essa tendência é devida em grande parte ao fato de que, enquanto o rápido desenvolvimento tecnológico dos equipamentos de computação levou à disponibilidade de computadores cada vez mais potentes a preços cada vez menores, a tecnologia de *software* não teve em modo algum um desenvolvimento comparável. De fato, a maior parte dos programadores usa hoje em dia ainda técnicas de programação da época dos computadores de válvulas.

Isso leva a que o processo de construção de programas seja muito ineficiente e, portanto, custoso. Os programas mesmos, feitos muitas vezes *ad hoc* e sem uma verdadeira metodologia, são geralmente cheios de erros, que somente através de um longo e novamente custoso processo de depuração são reduzidos a um nível "razoável".

Vemos então que os altos custos do desenvolvimento de programas devem-se principalmente à baixa produtividade dos programadores, a qual, por sua vez, é devida em grande parte à falta de uma metodologia de desenvolvimento de programas.

1.3.4.1 Confiabilidade X Complexidade

A confiabilidade de um produto é uma medida de sua capacidade de suportar o manuseio dentro de suas especificações de uso, mantendo-se capaz de atender às suas especificações de funcionamento. Naturalmente, quanto mais crítica a posição de um produto dentro de um sistema, mais confiabilidade lhe será exigida pelo usuário. A crescente dependência da sociedade em relação aos sistemas de computação faz com que a confiabilidade de programas seja uma questão crítica em muitas situações e ocasiões.

Ora, a confiabilidade de um agregado de componentes depende tanto da confiabilidade dos componentes isolados como da confiabilidade de sua interação. No caso dos produtos de *software*, os componentes elementares, isto é, as instruções, são tão confiáveis quanto o *hardware* da máquina que os executa. No atual estado da tecnologia de sistemas digitais, esta confiabilidade é sempre muito alta, desde que se disponha de suporte adequado de manutenção; e, em caso de necessidade, há maneiras de configurar os sistemas de computação de modo a torná-los tão confiáveis quanto se queira (pagando o respectivo preço, é claro). De qualquer maneira, o programador não precisa normalmente se preocupar com a confiabilidade da máquina: em geral, quando ocorre erro de máquina, os sintomas são tais que é muito fácil identificar o culpado.

Já a outra causa de falhas, a complexidade das interações, oferece, no caso dos programas, campo muito fértil à produção de defeitos; em razão das próprias e já destacadas dimensões da complexidade do *software*. Programas de aplicação, em sistemas de computação de médio porte, contêm tipicamente da ordem 10^4 a 10^5 instruções de máquina, selecionadas de repertório da ordem de algumas centenas de instruções (consideradas todas as modificações possíveis nos campos das instruções). Como a transferência de controle entre as instruções funcionará sempre corretamente se não houver defeito de *hardware*, os possíveis erros serão sempre causados

por uma escolha errônea das instruções. Para programas com as dimensões aqui estimadas, escrever um programa particular pode significar escolher um dentre 10^{100000} programas! Sabendo-se que o número de átomos do universo, pela estimativa mais exagerada, está longe de 10^{100}, o programador pode ter idéia de suas chances de acertar ao acaso.

Uma vez constatado, em fins dos anos 60, que as sistemáticas habitualmente usadas pelos programadores (inclusive o uso das ferramentas disponíveis) eram os grandes responsáveis pela baixa confiabilidade dos programas, surgiu a Programação Estruturada (PE). Ela se propõe, entre outras coisas, a fornecer sistemáticas gerais para construção de programas cuja confiabilidade pode ser verificada *a priori*, eliminando, idealmente, a etapa de depuração; isto é, *embutindo a confiabilidade no próprio projeto do programa.*

Por que a deficiência em confiabilidade da programação tradicional? Uma razão muito forte é que os programas produzidos pelos métodos tradicionais não evidenciam com muita clareza a relação entre o aspecto estático (o texto) e o aspecto dinâmico (a execução). Isto é, pode ser bastante difícil, pelo exame do texto de programa, prever o que acontecerá se ele for executado. Esta a razão pela qual a PE adotou como uma de suas bandeiras a eliminação do GOTO dos programas: para *remover um empecilho à identificação entre os aspectos estático e dinâmico do programa*, e não porque o uso do GOTO seja pecado.

1.3.4.2 Manutenibilidade

Para quem usa um programa, um item de custo nem sempre evidente à primeira vista é o prejuízo causado por erros não detectados durante a depuração. Nos programas oferecidos comercialmente, tornou-se prática comum lançá-los no mercado incompletamente depurados, esperar as reclamações dos usuários, descobrir os erros por análise das queixas e produzir novas versões à medida que os erros vão sendo corrigidos. Costuma ocorrer que em muitos produtos (principalmente sistemas operacionais), as novas versões venham com ampliações, que por sua vez contêm novos erros. Pode-se chegar a situações (registradas na literatura) em que o produto depois de um tempo tende a manter constante a taxa de erros, ou mesmo a aumentá-la.

Como se vê, os programas-produtos disponíveis comercialmente costumam sofrer revisões periódicas (*releases*), tanto para correção de erros detectados como para mudanças de especificações (freqüentemente, para oferecer melhores serviços que, por razões econômicas, não foi possível incluir nas versões anteriores). A esta atividade de revisão se chama de *manutenção do software*. A curva intermediária do gráfico mostra a tendência da proporção dos custos de manutenção comparados com os de desenvolvimento inicial para projetos de grande porte. Mesmo sendo difícil definir em forma exata quando termina o desenvolvimento de um programa, para passar à fase de manutenção, fica claro que a manutenção representa um custo cada vez mais importante no ciclo de vida de um programa.

A PE, em primeiro lugar, pode facilitar a manutenção simplesmente tornando-a menos freqüente, por conseguir aumentar a confiabilidade do produto original. Por outro lado, mesmo que se admita a inevitabilidade da permanência de alguns erros, a presença de estrutura no programa é sempre um auxílio valioso para sua detecção, por torná-lo muito mais legível e fácil de depurar.